Math Logic, Week#XII (*i.e.*, Week#C_{16})

Moon's Day — April Fools' Day + 2

Review Questions (for me to ask & you to answer)**:**

1. **¿What does the notation $\mu\, x_{k+1}\, R(x_1, \ldots, x_k, x_{k+1})$ mean?**
2. **¡State a theorem about the μ operator and recursive functions!**
3. **¿How did we code sequences of integers by integers?**
4. **¡Name some functions on those sequences that are recursive!**
5. **¿How did we encode formulas with integers?**
6. **¿Is the set of (Gödel numbers of) consequences of a finite set of axioms recursive?**
7. **¡Define: "a relation on $\mathbb{N}$ is representable in $\mathbb{A}_E$"!**
8. **¿What do we know about the structure of models of $\mathbb{A}_E$?**

Reminder: Homework Problem E:

1. Suppose R is a recursive set (unary relation) and that $f_1, \ldots, f_n$ are Σ_1-definable functions. (The f_i's may well have different numbers of argument places; say that f_i is k_i-ary.) The *closure* of R under the f_i's is the smallest set S such that (1) $R \subseteq S$ and (2) for each f_i and each $x_1, \ldots, x_{k_i} \in S$, $f_i(x_1, \ldots, x_{k_i}) \in S$. This set S can be constructed recursively as follows:

$$\begin{aligned} S_0 &= R \\ S_{i+1} &= S_i \cup \bigcup_{1 \le i \le n} \{f_i(x_1, \ldots, x_{k_i}) : x_1, \ldots, x_{k_i} \in S_i\} \\ S &= \bigcup_{i \in \mathbb{N}} S_i \end{aligned}$$

   Prove that S is Σ_1-definable (i.e., r.e.).

2. Suppose that, in addition, the functions $f_1, \ldots, f_n$ are *increasing*; that is, whenever $f_i(x_1, \ldots, x_{k_i}) = y$, $y > x_1$, $y > x_2$, …, and $y > x_{k_i}$. Prove that the set S above is then recursive.

Incompleteness: (Reminder: I write $\mathfrak{N}_E$ where Enderton writes $\mathfrak{N}$)

1. In this section we continue to assume that we are working in the language of $\mathfrak{N}_E$. Generalizations are possible.
2. **Fixed Point Theorem:** For any formula β in which only v_1 occurs free, there is a sentence σ such that

$$\mathbb{A}_E \models (\sigma \leftrightarrow \beta(S^{\#(\sigma)}0)).$$

Discussion, quoted from Enderton, page 227:

> *We can think of σ as* indirectly *saying, "β is true of me." Actually σ doesn't say anything of course, it's just a string of symbols. And even when translated into English according to the intended structure $\mathfrak{N}_E$, it then talks of numbers and their successors and products and so forth. It is only by virtue of our having associated numbers with expressions that we can think of σ as referring to a formula, in this case to σ itself.*

Proof: **[slightly expanded from Enderton]**

Let $\theta(v_1, v_2, v_3)$ be a formula representing the function

$$sb(v_1, v_2) = \begin{cases} \#(\alpha(S^{v_2}0)) & \text{if } v_1 = \#(\alpha) \text{ for some } \alpha \\ & \text{with (at most) 1 free variable} \\ 0 & \text{otherwise.} \end{cases}$$

Trick #1: Consider the formula

$$\forall v_3(\theta(v_1, v_1, v_3) \rightarrow \beta(v_3)). \tag{1}$$

> **Note that** we have set the first two argument positions of θ to the same variable. [So, essentially,] the Gödel-number v_1 of a formula is being substituted into that formula, a process called "self reference." Enderton points out that "*This formula has only v_1 free. It defines in $\mathfrak{N}_E$ a set to which $\#(\alpha)$ belongs iff $\#(\alpha(S^{\#(\alpha)}0))$ is in the set defined by β .*"

Let q be the Gödel-number of formula (1).

Reminders:

- $\theta(v_1, v_2, v_3)$ "says" $v_3 = \begin{cases} \#(\alpha(S^{v_2}0)) & \text{if } v_1 = \#(\alpha) \text{ for some } \alpha \\ & \text{with (at most) 1 free variable} \\ 0 & \text{otherwise.} \end{cases}$
- q is the Gödel number of $\forall v_3(\theta(v_1, v_1, v_3) \rightarrow \beta(v_3))$ (*i.e.*, formula 1).

Trick #2: Replace v_1 in formula (1) with S^q0; the result is a

$$\sigma \;=\; \forall v_3(\theta(S^q0, S^q0, v_3) \rightarrow \beta(v_3)). \tag{2}$$

Note: Replacing v_1 with q is more self reference. Sentence σ asserts (in $\mathfrak{N}_E$) that $\#(\sigma)$ is in the set defined by β.

Finally, for this σ we shall prove our original claim that

$$\mathbb{A}_E \models (\sigma \leftrightarrow \beta(S^{\#(\sigma)}0)). \tag{3}$$

Recall that, since q is in fact a natural number, in every model of $\mathbb{A}_E$ there is a unique element x which satisfies $\theta(S^q(0), S^q(0), x)$ — the actual Gödel-number of the substituted formula. How do we create that formula? By definition of θ:

take the formula with Gödel-number q: $\forall v_3(\theta(v_1, v_2, v_3) \rightarrow \beta(v_3))$
and substitute S^q0 for v_1: $\forall v_3(\theta(S^q0, S^q0, v_3) \rightarrow \beta(v_3))$

That formula is σ itself — the substitution operation described by the formula is just the operation we made in defining σ itself! So the Gödel-number of the resultant formula is $\#(\sigma)$. Thus the only element v_3 which can satisfy $\theta(S^q, S^q, 0)$ is $\#(\sigma)$ itself, or, more precisely,

$$\mathbb{A}_E \models \forall v_3(\theta(S^q0, S^q0, v_3) \;\leftrightarrow\; v_3{=}S^{\#(\sigma)}0). \tag{4}$$

Reminders:

- σ is the formula $\quad \forall v_3(\theta(S^q0, S^q0, v_3) \rightarrow \beta(v_3))$.
- Assertion (3) was $\quad \mathbb{A}_E \models (\sigma \leftrightarrow \beta(S^{\#(\sigma)}0))$.
- Assertion (4) was $\quad \mathbb{A}_E \models \forall v_3(\theta(S^q0, S^q0, v_3) \leftrightarrow v_3 = S^{\#(\sigma)}0)$.

Now we prove assertion (3):

$\Rightarrow$: Trivially, from the statement of σ,

$$\sigma \models \theta(S^q0, S^q0, S^{\#(\sigma)}0) \rightarrow \beta(S^{\#(\sigma)}0).$$

By statement 4, $\mathbb{A}_E \models \theta(S^q0, S^q0, S^{\#(\sigma)}0)$.
Thus $\mathbb{A}_E \cup \{\sigma\} \models \beta(S^{\#(\sigma)}0)$.

$\Leftarrow$: Statement (4) implies that

$$\beta(S^{\#(\sigma)}0) \rightarrow [\forall v_3(\theta(S^q0, S^q0, v_3) \rightarrow \beta(v_3))].$$

The part in the square brackets is just σ.

3. **Tarski's Theorem on the Undefinability of Truth:** (for $\mathfrak{N}_E$):
The set $\#(Th(\mathfrak{N}_E))$ is not definable in $\mathfrak{N}_E$.

Proof: Suppose it were definable, say by some formula $\beta(x)$. By the fixed point theorem, *applied to* $\neg\beta$, there is a sentence σ such that

$$\mathfrak{N}_E \models (\sigma \leftrightarrow \neg\beta(S^{\#(\sigma)}0)).$$

But then σ is true in $\mathfrak{N}_E$ iff $\mathfrak{N}_E \models \neg\beta(S^{\#(\sigma)}0)$, contradicting the assumption about β.

Historical Note: This is just a version of the liar's paradox. If β did in fact define the set of statements true in $\mathfrak{N}_E$, σ would "say" "I am false."

4. **Corollary:** *$\#(Th(\mathfrak{N}_E))$ is not r.e.*

5. **Corollary: Gödel's Incompleteness Theorem [1931]:**
   If $A \subseteq Th(\mathfrak{N}_E)$ and A is recursive, then $Cn(A)$ is not complete.
   Corollary: $Th(\mathfrak{N}_E) \neq Cn(\mathbb{PA}_E)$.

6. **Lemma:**

   If $Cn(\Sigma)$ is recursive then $Cn(\Sigma \cup \{\gamma\})$ is recursive for any sentence γ.

   Proof: $\Sigma \cup \{\gamma\} \models \alpha$ *iff* $\Sigma \models \gamma \rightarrow \alpha$. Thus $\#(\alpha) \in Cn(\Sigma \cup \{\gamma\})$ iff $\#(\gamma \rightarrow \alpha) \in Cn(\Sigma)$. Since the function mapping each $\#\alpha$ to $\#(\gamma \rightarrow \alpha)$ is recursive (by Theorem concatenation).

7. **Thrm: [Strong undecidability of $Cn(\mathbb{A}_E)$]**

   Let T be a theory such that $T \cup \mathbb{A}_E$ is consistent. Then $\#(T)$ is not recursive.

8. **Church's Theorem**

   The set of logically valid sentences of first order logic with equality is not recursive.

   (This holds true in any language which is rich enough — by our proof, any language at least as rich as the language of $\mathfrak{N}_E$: thus any language containing at least one constant symbol, at least 4 function symbols, of which 3 are at least binary (i.e., take at least two arguments), and one relation symbol which is at least binary.)

9. **Gödel's 2nd Incompleteness Theorem** [beyond the scope of this course]

   ?'What does the following formal statement about $\gamma(\Gamma)$ assert?

   $$\neg\exists\#\phi\exists P(P \text{ encodes a deduction of } (\phi \wedge \neg\phi) \text{ from } \Gamma)$$

   ¡$\mathbb{PA}_E \models \gamma(\mathbb{PA}_E)$ if and only if $\mathbb{PA}_E$ is inconsistent!

   (And analogously for all "stronger" theories.)

Temporal Logic & Model Checking

1. **Change of approach:** Now we start by *fixing a model* and ask whether the model has some — often complicted — properites.
   - We'll look (briefly) at 2 such logics.
   - What we'll do here is related to verification that chips or simple systems meet their specification.

     (I say "*simple systems*" since we're going to go back to propositional logic, which has severe limitations on expressibility.)
   - **Limit attention** to "*reactive systems*" — systems that are intended to run forever and respond to environmental stimuli.

     . . . rather than systems intended to make a computation and stop.

     (Such systems include operating systems, embedded systems, and computer hardware.)

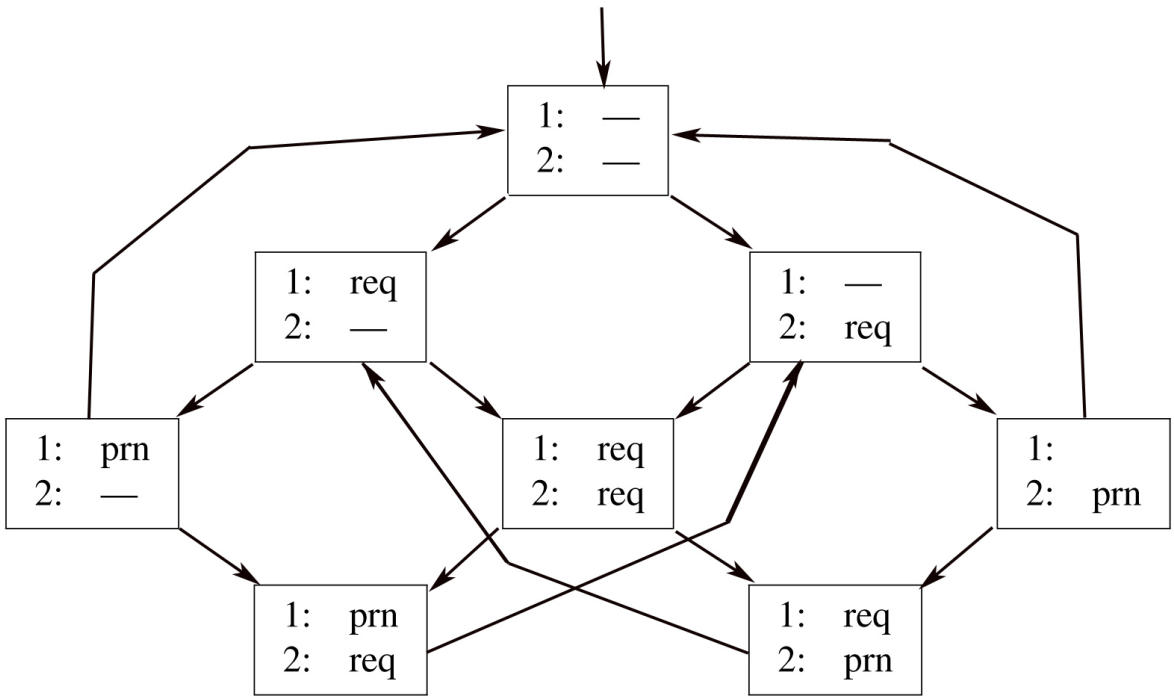
     Seems especially helpful with *concurrency*.
   - **Formal Issue:** To study such issues, we need a logic of things changing over time — a "*temporal logic*."
     It's important that the logic be:
     - expressive enough to express interesting properties, but
     - inexpressive enough that we can compute the properties.
   - **General Practical Issue for formal verification:**
     Detailed correctness proofs are, in general, very expensive
     . . . but errors can be even more expensive.

     (And, of course, if our systems had <u>no</u> errors, maybe there'd be no market for specialists in malware avoidance.)

2. An example of a model for 2 computers, 1 and 2, sharing a printer [from Huth & Ryan 2004]:

   - The states represent how each computer is interacting, at one time, with the printer.
   - At any time, each computer may be printing (prn), requesting access to the printer (req), or doing something non-critical (—).
   - The arrows represent how those states may change due to one action.
   - Note that the model does not deal with one sticky issue: what happens if it seems as if 2 actions happen at exactly the same time.
   - Requirement: every state has at least one edge out (since a reactive system is supposed to run forever).

The model — here, of a protocol



3. Syntax of this temporal logic:
   - Proposition letters — in our example, n_i, r_i, and p_i (for $i = 1, 2$).
   - Enough propositional connectives — say $\neg$, $\vee$, $\wedge$, and $\rightarrow$.
   - Often $\top$ (true) and $\bot$ (false)
   - Temporal operators, often including

Notation:	**Rough semantics**
$(X\varphi)$	φ will be true at the next state.
$(F\varphi)$	φ will be true at some present or future state.
$(G\varphi)$	φ will be true at all present or future states.
$(\varphi U\psi)$	φ will be true until, sometime, ψ becomes true.

4. **¿But what does this all mean?** We'll mention 2 samples:

LTL (*Linear-time temporal logic*)**:** To evaluate a formula φ:

(a) First fix an infinite path through the graph of states — call the sequence of states $s_1, s_2, s_3, \ldots$. This represents some possible sequence of events.
   - By induction on subformulas of φ,
     evaluate each subformula at each state s_i according to path $s_i s_{i+1}, s_{i+1}, \ldots$.
   - Then see whether φ evaluates to true at s_1.

(b) And then see whether φ is satisfied on all paths through the graph.

Since there are infinitely many paths through most such state diagrams,
computing satisfaction in LTL is very computationally complex.
¡But it can be computed in finite, albeit large, time!

CTL (*Computation tree logic*)**:**

(a) Each of the temporal operators is (*and must be*) preceded by a modifier:

A for all paths starting here

E for some path starting here

(b) Again, to evaluate a formula at a state φ,
by induction on subformulas ψ,
evaluate ψ at every state.

(c) Evaluate based upon all possible routes out of every state at every step.
So at each step you consider all possible next steps — there's no way to predict choices more than one step in advance.

(d) This computation turns out to be a lot easier than the one for LTL.

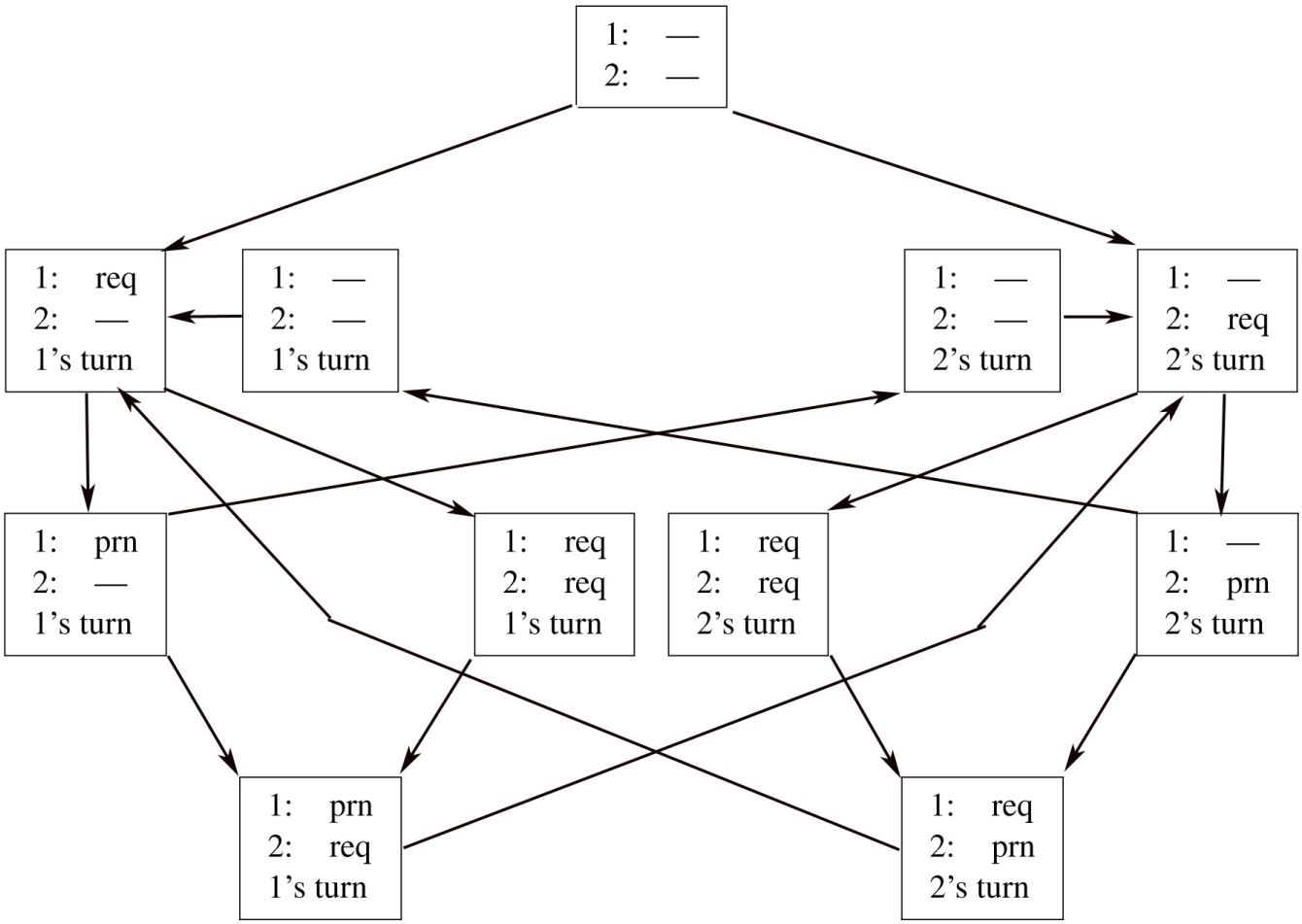
5. **Example property: Liveness.** *Whenever any process requests to use the printer, it will eventually be permitted to do so.*

¿Can we say this (specifically: in LTL, and/or in CTL)**?**

- If computer i is currently requesting the printer, it will ultimately get access:
  In LTL: $r_i \to Fp_i$
  In CTL: $r_i \to AFp_i$
- Whenever either computer requests the printer, it will ultimately get access:
  In LTL: $G(r_1 \to Fp_1) \wedge G(r_2 \to Fp_2)$
  In CTL: $AG(r_1 \to AFp_1) \wedge AG(r_2 \to AFp_2)$

¿Are these formulas true in the model?

6. Revised model protocol



- **¿Does this model satisfy the liveness properties?**
  In LTL: $G(r_1 \rightarrow Fp_1) \wedge G(r_2 \rightarrow Fp_2)$
  In CTL: $AG(r_1 \rightarrow AFp_1) \wedge AG(r_2 \rightarrow AFp_2)$

- **¿Do you see any other disadvantages?**

7. Problem was *strict sequencing*: We took turns between computers, to the point that, when it was 2's turn to print, we refused to print anythng for computer 1 ...

   ... even if 2 has not submitted anything to print.

   ¿How can we say there is no strict sequencing?

   - 1st describe an instance of strict sequencing: In LTL:

     $$p_1 \wedge \neg r_2 \wedge X((\neg p_1 U p_2) \vee G \neg p_1)$$

     (or, of course, the same with the computers reversed)

   - We want to say that situation is to be *avoided*:
     That statement is fairly obvious in CTL:

     $$AG((p_1 \wedge \neg r_2) \rightarrow EX(E(\neg p_2 U p_1))) \\ \wedge \\ AG((p_2 \wedge \neg r_1) \rightarrow EX(E(\neg p_1 U p_2))).$$

     (In CTL, $AG\varphi$ says that φ will always be true at every state.)

   Now, ¿how can we modify the model to avoid strict sequencing?